МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет имени
первого Президента России Б. Н. Ельцина»

МАКРОСТАТИСТИЧЕСКИЙ АНАЛИЗ И ПРОГНОЗИРОВАНИЕ ДАННЫХ

Лекция № 9

Ассимиляция данных и методы коррекции

Екатеринбург
2024

Содержание

Часть 1. Прогнозирование на основе метода «Гусеница»

В методе сингулярного спектрального анализа (SSA) временному ряду ставится в соответствие набор векторов, составленных из скользящих отрезков ряда выбранной длины L. Эти вектора порождают траекторные матрицы, к которым можно привязать различные линейные рекуррентные соотношения, на основе которых потом и строится прогноз.

Таким образом, при прогнозировании с помощью метода SSA рассматривается множество временных рядов, описываемых с помощью **линейных рекуррентных формул** (ЛРФ). При этом порядок ЛРФ может быть заранее неизвестен, но это и не требуется, если производить только прогноз. Ряд, управляемый ЛРФ, естественным образом порождает **рекуррентное продолжение**, так как каждый его член является линейной комбинацией некоторого количества предыдущих. Поэтому коэффициенты этой линейной формулы могут быть использованы и для продолжения ВР. Важно отметить, что необязательно искать ЛРФ минимальной размерности, так как любая ЛРФ, управляющая рядом, приводит к одному и тому же продолжению, а значит, метод можно считать адаптивным.

Общая идея нахождения ЛРФ состоит в следующем. Пусть d – минимальная размерность всех ЛРФ, управляющих рядом. Если длина окна L больше, чем d, число отсчетов ряда достаточно велико, то размерность траекторного пространства равняется d. Это траекторное пространство порождает ЛРФ размерности $L-1$, которая управляет ВР. Исходный базовый метод SSA порождает базис траекторного пространства, а уже на его основе строится управляющая ЛРФ. Применяя ЛРФ к последним точкам ряда, будет получено продолжение ВР, то есть его прогноз. В условиях асимптотической разделимости, продолжение будет строиться для всех компонент, составляющих данный временной ряд, поэтому и декомпозировать его нет необходимости.

С траекторными линейными пространствами и порождаемыми ими линейными рекуррентными формулами связана целая отдельная теория. Здесь мы не станем ее подробно рассматривать, а будем только использовать результаты, уже полученные и доказанные в этой области науки.

Опишем формально алгоритм рекуррентного прогнозирования.

1) Входными данными **алгоритма прогнозирования на основе SSA** являются: исходный временной ряд $F_N = (f_0,..., f_{N-1})$ с числом отсчетов *N*, выбранная длина окна *L*, число *M* точек прогноза.
2) Определим некоторое линейное пространство $\Im$ размерности $r < L$. Обычно это пространство задается некоторым ортонормированным базисом $P_1,...P_r$, но результаты прогнозирования не зависят от вида конкретного базиса.
3) По алгоритму SSA мы можем построить траекторную матрицу заданного ВР $\mathrm{X} = [X_1 : ... : X_K],\ K = N - L + 1.$
4) Найдем матрицу $\dot{X} = \sum_{i=1}^{r} P_i P_i^T \mathrm{X}$, где $P_1,...P_r$ - ортонормированный базис пространства $\Im$. По сути, новые вектора траекторной матрицы будут ортогональной проекций X на пространство $\Im$.
5) Так как новая матрица будет $\hat{\mathrm{X}}$ ганкелевой (или будет преобразована к таковой), то по теории сингулярного спектрального анализа он является траекторной матрицей некоторого нового временного ряда $\tilde{F}_N = (\tilde{f}_0,..., \tilde{f}_{N-1})$.
6) Пусть $Y = (y_1,..., y_L)^T$ – вектор из пространства $\Im$. В теории ЛРФ доказывается, что последняя координата этого вектора является линейной комбинацией его первых компонент:

$$y_L = a_1 y_{L-1} + a_2 y_{L-2} + ... + a_{L-1} y_1 \qquad (9.1)$$

7) Положим

$$v^2 = \pi_1^2 + \pi_2^2 + ... + \pi_r^2 \tag{9.2}$$

где π_i – последняя компонента базисного вектора P_i.

8) Тогда ряд весовых коэффициентов $R = \left(a_{L-1},...,a_1\right)^T$ может быть найден по формуле:

$$R = \frac{1}{1-v^2}\sum_{i=1}^{r}\pi_i P_i^{\nabla} \tag{9.3}$$

и не зависит от выбора базиса $P_1,...P_r$ пространства $\Im$.

9) Используя приведенные выше обозначения и формулы, прогноз на основе метода SSA с учетом ЛРФ есть ВР $G_{N+M} = \left(g_0,...,g_{N+M-1}\right)$, который строится как:

$$g_i = \begin{cases} \tilde{f}_i & ,i < N \\ \sum_{j=1}^{L-1} a_j g_{i-j}, & i = N,...,N+M-1 \end{cases} \tag{9.4}$$

Вторая строка в (9.4) как раз и есть прогнозные отсчеты ВР.

Подобная методика прогноза на основе метода SSA и использовании ЛРФ имеет свои преимущества и недостатки. Преимуществами такого подхода являются его строгое математическое описание, широкие возможности вариации начальных параметров и отсутствие необходимости описка оптимальной группировки компонент. Но у метода есть и существенный недостаток – он плохо формализуем в виде программных алгоритмов. В самом деле, один из входных параметров (базис пространства) необходимо определять случайным образом, но при этом таким образом, чтобы не нарушить его характерные свойства. Есть специализированные алгоритмы построения подобных векторов, но они требуют анализа тех ВР, на которые будут опираться, из-за чего преимущество адаптивности прогноза будет потеряно.

В связи с этим, чаще на практике вместо подобного строгого теоретического описания прогноза на основе метода SSA в комбинации с ЛРФ используется, так называемое, **статистическое SSA-прогнозирование**.

Методика этого прогноза выглядит следующим образом:

1) Входными данными являются: исходный временной ряд $F_N = (f_0, ..., f_{N-1})$ с числом отсчетов N, выбранная длина окна L, число M точек прогноза.
2) Проведем *частичный* SSA-анализ заданного ВР: только этап разложения и, самое главное, определим **метод группировки** I главных компонент (то есть определим номера группируемых собственных троек). **Усреднения собственных троек не происходит**.
3) К ВР F_N добавляется всего **один случайный отсчет** из диапазона уже имевшихся уровней ряда.

$$F'_{N+1} = (f_0, ..., f_{N-1}, f_{new}),\ f_{new} \in [\min(F_N); \max(F_N)] \qquad (9.5)$$

4) Этот ряд длины N+1 подвергается SSA-декомпозиции (**только** шаги разложения и формирования траекторной матрицы), но **без какой-либо оценки параметров**, а на основе оценок построенных уже на 2 шаге алгоритма.
5) Полученные собственные тройки нового ряда F'_{N+1} **группируются и усредняются** на основе метода группировки I из 2 шага данного алгоритма. Важно, никаких новых оценок не происходит.
6) В результате усреднения будет получен новый временной ряд $G_{N+1} = (f_0, ..., f_{N-1}, g_N^{(i)})$, для которого первые N отсчетов совпадают с исходным ВР, а последний отсчет является **пред-прогнозом** . Первые отсчеты ряда G не изменяются в сравнении с исходным рядом, так как ни траекторная матрица, ни способ группировки, ни другие параметры в ходе работы алгоритма не изменялись.

7) Отсчет $g_N^{(i)}$ является лишь **первичным приближением прогноза**. Для получения точного прогноза, новый отсчет $f_{new} = g_N^{(i)}$ ряда F'_{N+1} приравнивается этому приближению, после чего **шаги с 4 по 6 повторяются** до тех пор, пока разница $g_N^{(i)} - g_N^{(i-1)}$ не станет близка к нулю, то есть пока **значение** $g_N^{(i)}$ **не перестанет изменяться** с повторением шагов 4-6.
8) Полученный в результате отсчет $g_N^{(i)}$ принимается за первую точку прогноза. Для продолжения прогноза, новый ряд длины N+1 становится ВР для прогнозирования $F_{N'} = G_{N+1}$, и алгоритм **повторяется вновь,** но только с третьего шага, так как на протяжении всего алгоритма прогноза нет необходимости заново искать необходимую группировку компонент.

Как видно из описания подобного алгоритма, он во многом эквивалентен прогнозу на основе ЛРФ пространств. Но подобное описание гораздо проще реализуется в виде прикладных программ.

У данного алгоритма прогнозирования тоже есть свои преимущества и недостатки. Недостатком алгоритма является необходимость выбора метода группировки компонент *I*. Это не всегда простая задача, более того, точность прогноза больше всего зависит именно от выбора группировки компонент, нежели чем от других каких-либо параметров. На практике, для прогнозирования нужно группировать **как можно меньше собственных троек**, так как чем больше их мы сгруппируем, тем хуже будет согласована конечная траекторная матрица для диагонального усреднения, и тем быстрее может разбежаться конечный прогноз. Другая проблема, связанная с группировкой, состоит в том, что если мы восстанавливаем не только главные компоненты (тренд, сезоны, циклы), но и шумовые, то точность прогноза оказывается не адекватной. Еще одна неочевидная проблема данного

алгоритма – это его скорость. Шаги с 4 по 6 повторяются вновь и вновь, пока не перестанет меняться прогнозируемая точка. Дело в том, что некоторые такие точки имеют **очень медленную сходимость**. В результате число повторений этих шагов может достигать огромного числа без существенного увеличения точности. В этом случае обычно добавляют ограничение на число итераций цикла шагов 4-6, после которых либо прерывают прогноз, либо принимают за прогноз последнюю найденную точку.

Достоинствами алгоритма являются простота его реализации, высокая степень автоматизации и адаптивности, а также тот факт, что все параметры метода (включая способ группировки собственных троек) за весь алгоритм прогноза мы определяем **только один раз**. Нам нет необходимости после каждой новой точки прогноза переанализировать ряд, хотя это и можно делать в целях повышения точности прогноза, подобно схемам коррекции прогноза на основе авторегрессий.

Часть 2. Прогнозирование на основе нейронных сетей

В настоящее время искусственные нейронные сети (далее, просто нейронные сети) используется для решения самого широкого класса задач, начиная от распознавания образов и классификации до создания систем искусственного интеллекта. Было бы удивительно, если бы рано или поздно не появилось методики использования нейронных сетей для анализа и прогнозирования временных рядов. В данном случае нас больше всего интересует задача **прогнозирования ВР с помощью нейронных сетей**, нежели задача их анализа и декомпозиции. Именно в этой области у нейронных сетей есть существенные преимущества, в первую очередь связанные с тем, что их использование для прогноза делает его **адаптивным**, то есть такое прогнозирование полностью опирается только на сами отсчеты ВР и не требует его предварительной декомпозиции и подробного анализа.

Искусственные нейронные сети представляют собой математическую модель сети биологических нервных клеток, описываемую в виде системы соединенных и взаимодействующих простых узлов – нейронов (рис. 1). Каждый из этих узлов только принимает и посылает сигналы. Их конфигурация при этом может существенно отличаться (как и число слоев).

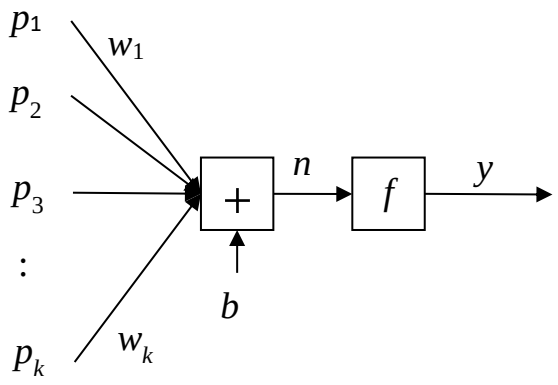

Рисунок 9.1 – Отдельный нейрон и его конфигурация

Представленный на рисунке 9.1 нейрон имеет k входов p_k, смещение b (*bias*), весовые коэффициенты w_k, суммарный выход сети n, функцию активации/перехода f и выход y. Для такой модели нейрона выход y рассчитывается просто как:

$$y = f\left(\sum_{i=1}^{k} w_i p_i + b\right)$$

Если функция активации есть просто линейная функция, то выход нейрона представляет собой простую линейную взвешенную сумму его входов (плюс смещение).

Входами p_k, как мы увидим дальше, могут служить различные исходные данные, выходы других нейронов, а также и сам выход этого нейрона (обратная связь). Смещение b, чаще всего, равняется 1 (b = 1), чтобы обеспечить удаленность взвешенной суммы от нулевого значения. Весовые коэффициенты w_k являются приспособляемыми и регулируемыми параметрами нейрона, то есть именно их и калибрует каждый отдельный нейрон в ходе процесса обучения нейронной сети. Веса для одного нейрона образуют одномерный вектор значений $\{w_1, w_2, ..., w_k\}$, а для целой сети нейронов – матрицу весовых коэффициентов $\boldsymbol{W}$. При этом выражение ранее можно будет переписать в векторно-матричной форме как $y = f(\mathbf{W}\mathbf{p} + b)$.

Функция активации или функция перехода f может быть любой линейной или нелинейной функцией в зависимости от типа решаемых задач. Укажем только несколько ключевых из них. Функция ограничителя с резким порогом (hard limit transfer function) дает 0, если ее аргумент меньше нуля, и дает 1, если аргумент больше либо равен нулю (рис. 9.2a). Как вариант, иногда эту функцию изменяют на пределы от -1 до 1 или других констант. Линейная функция (рис. 9.2b) просто передает взвешенную сумму входов без изменений $y = \sum w_i p_i + b$. Сигмоидная (точнее, лог-сигмоидная) функция активации (рис. 9.2c) определяется формулой: $y = f(n) = \dfrac{1}{1+e^{-n}}$.

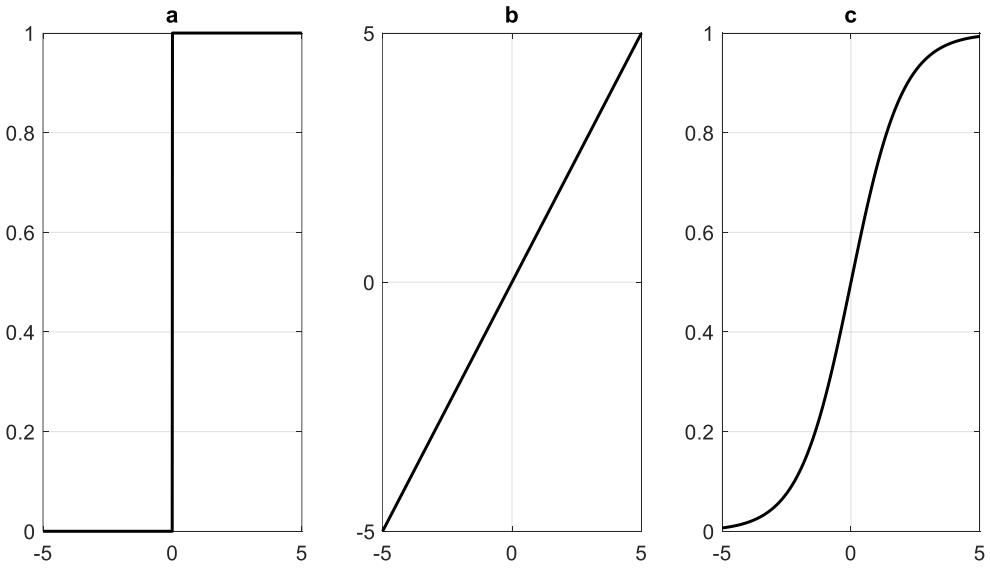

Рисунок 9.2 – Основные функции активации, при b=0: a – функция ограничителя с резким порогом, b – линейная, c – лог-сигмоидная функция

Одиночный слой нейронов, состоящий из S базовых нейронов, представлен на рисунке 9.3. Стоит отметить, что каждый из входов p_k подается на каждый нейрон. В самом деле, если эти входы для нейрона будут не важны, то он просто обнулит их весовые коэффициенты.

В этой схеме выходы y_k могут объединяться в общий выход с помощью некоторой функции (или же объединяться на уровне n_k), а могут служить входами для следующего слоя нейронов. При этом число входов R чаще всего не совпадает с числом нейронов S в заданном слое. Для такого одиночного слоя нейронов весовые коэффициенты задаются матрицей **W**:

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

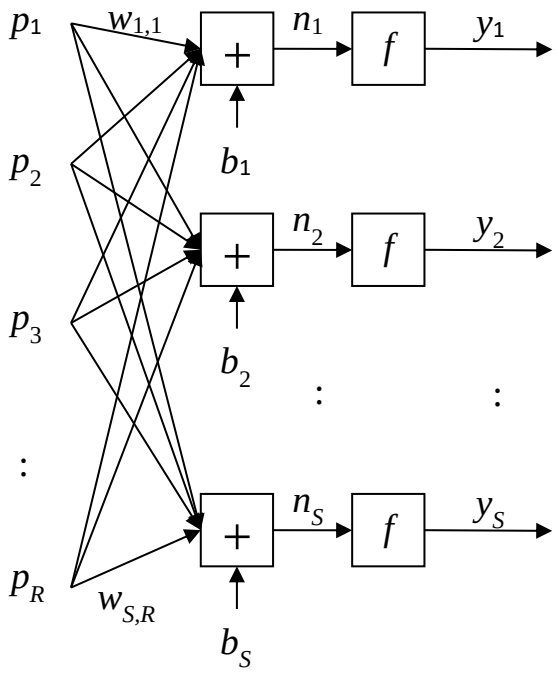

Рисунок 9.3 – Одиночный слой из *S* нейронов

Аналогичным образом, для слоя выражение зависимости для выхода нейронного слоя будет выглядеть в матричной форме как $y = f\left(\mathbf{Wp} + \mathbf{b}\right)$.

Рисунок 9.3 является достаточно громоздким, поэтому в сокращенной форме его можно свести к изображению на рисунке 9.4, называемым одним простым слоем нейронов.

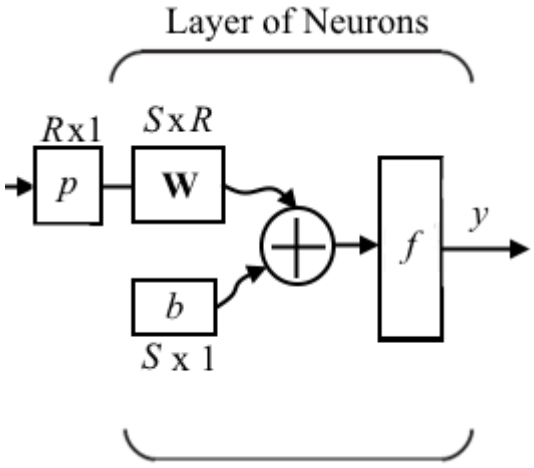

Рисунок 9.4 – Одиночный просто слой из *S* нейронов на *R* входов

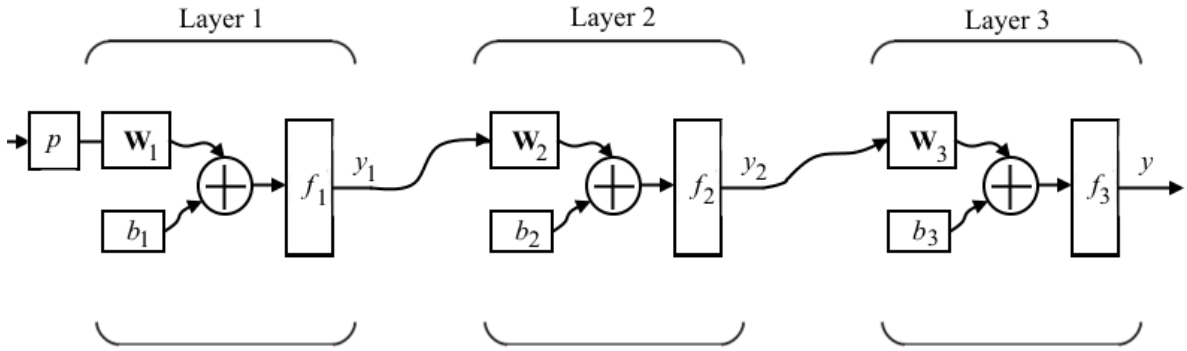

Рисунок 9.5 – Нейронная сеть из 3 слоев

Для создания многослойной нейронной сети выходы одного слоя подают на входы последующего, образуя сложную структуру связи отдельных нейронов и функций активации. Например, на рисунке 9.5 представлена трехслойная нейронная сеть, для которой суммарный выход определяется выражением:

$$y = f_3\left[\mathbf{W}_3 f_2\left\{\mathbf{W}_2 f_1\left(\mathbf{W}_1\mathbf{p}+\mathbf{b}_1\right)+\mathbf{b}_2\right\}+\mathbf{b}_3\right]$$

Многослойные искусственные нейронные сети обладают значительно более мощными возможностями, чем простые однослойные, поэтому на практике обычно применяют именно их, хотя рост числа слоев приводит к существенному повышению вычислительной сложности соответствующей нейронной сети. Кроме представленной структуры связи нейронных слоев, бывают еще и дополнительные – на основе рекуррентных соотношений, наличие обратной связи, наличие интеграторов и т.д.

Как уже было сказано ранее, весовые коэффициенты **W** слоя нейронов являются параметрами соответствующей нейронной сети, которые мы и хотим установить/откалибровать в ходе процесса обучения искусственной нейронной сети. В случае процесса обучения с учителем, мы хотим при заданных входах, смещении и функции активации найти такие весовые коэффициенты, которые бы наилучшим образом давали известную выходную

функцию. То есть при обучении нейронной сети с учителем у нас имеется некоторое обучающее множество пар

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots \{\mathbf{p}_q, \mathbf{t}_q\}$$

где **p** – входы нейронной сети, которым соответствуют целевые выходы **t**.

При подаче на входы нейронной сети входов **p** мы получаем выходные значения, которые затем сравниваются с целями **t**. В зависимости от близости полученных результатов к желаемым происходит коррекция весовых коэффициентов – обычно по методу обратной связи, либо с помощью других способов обратного распространения ошибки. С математической точки зрения такой процесс обучения нейронной сети полностью соответствует многопараметрической задачи нелинейной оптимизации $y = \arg\min_{y \in A} \mathbf{W}(y, t)$.

Кроме обучения с учителем, существуют и другие принципы обучения, например, обучение без учителя (самоорганизующиеся сети) и обучение с подкреплением (на основе штрафов и поощрений). В данной работе они не рассматриваются.

Каким же образом калибровать сами весовые коэффициенты? Имеется множество методик (методы градиентного спуска, метод минимизации эмпирического риска и т.д.), но их общий принцип заключается в оценке значения ошибки *e* (*mse*, *sse* и любые другие) между полученным выходом нейронной сети *y* и целевым значением обучающей выборки *t*. Весовые коэффициенты, изначально заданные случайно или константой, затем изменяются пропорционально этой ошибке *e*, а весь алгоритм обучения повторяется до тех пор, пока не будет найден минимум отличия между выходом *y* и целевым значением обучающей выборки *t*.

Рассмотрим простой пример обучения нейронной сети, которая должна уметь распознавать, например, «яблоки» и «груши» (не в смысле распознавания их изображений или формы, а как принадлежность к двум различным множествам: {1} и {-1}). Для подобной задачи хватит однослойной нейронной сети с тремя входами и одним нейроном, и функцией активации ограничителя с резким порогом (рис. 9.4, $S = 1$, $R = 3$). Условимся, что если на выходе 1, то на входе были «яблоки» (единицы), если на выходе 0, то на входе были «груши» (минус единицы). Всю такую нейронную сеть можно математически описать как

$$y = f\left(\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + b\right)$$

где w – весовые коэффициенты (калибруем в процессе обучения), p – входные значения, b – смещение, f – функция активации ограничителя с резким порогом, y – выход нейронной сети.

Мы пытаемся откалибровать данную нейронную сеть с помощью обучения с учителем на следующей обучающей выборке.

$$\left\{p_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, t_1 = 0\right\}, \quad \left\{p_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, t_2 = 1\right\}$$

В этой выборке если на входе больше «яблок» (единиц), то выдаем единицу, если на входе больше «груш» (минус единиц) – выдаем ноль.

Ошибкой обучения считаем разность между целью и выходным результатом ($e = t - y$). Калибровка весовых коэффициентов и смещения тогда происходит с помощью следующих выражений:

$$w_i^{new} = w_i^{old} + e_i p,$$
$$b^{new} = b^{old} + e_i.$$

Обычно исходные весовые коэффициенты w и смещение b выбираются случайно. Пусть исходно они заданы следующими значениями:

$$\mathbf{W} = \begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix}, \quad b = 0.5$$

При таких значениях весовых коэффициентов на выходе нейронной сети для входа p_1 получится значение, равное 1:

$$y = f\left(\begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 0.5 \right) = f(2.5) = 1$$

Ошибка тогда равна $e = t_1 - y = 0 - 1 = -1$. Обновляем наши весовые коэффициенты и смещение:

$$w_i^{new} = w_i^{old} + e_i p = \begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} + (-1) \cdot \begin{bmatrix} 1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix},$$
$$b^{new} = b^{old} + e_i = 0.5 + (-1) = -0.5$$

На этом заканчивается первый шаг обучения. На втором шаге обучения мы берем уже весовые коэффициенты и смещение из нового вектора и вычисляем выход нейронной сети на второй обучающей выборке p_2:

$$y = f\left(\begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} - 0.5 \right) = f(-1.5) = 0$$

Ошибка тогда равна $e = t_2 - y = 1 - 0 = 1$. Согласно (9) обновляем наши весовые коэффициенты и смещение:

$$w_i^{new} = \begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix} + 1 \cdot \begin{bmatrix} 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 0.5 & 1 & -0.5 \end{bmatrix},$$
$$b^{new} = -0.5 + 1 = 0.5$$

Аналогично, третий шаг:

$$y = f\left(\begin{bmatrix}0.5 & 1 & -0.5\end{bmatrix}\begin{bmatrix}1\\-1\\-1\end{bmatrix} + 0.5\right) = f(0.5) = 1$$

Ошибка тогда равна $e = t_1 - y = 0 - 1 = -1$. Обновляем наши весовые коэффициенты и смещение:

$$w = \begin{bmatrix}0.5 & 1 & -0.5\end{bmatrix} - 1 \cdot \begin{bmatrix}1 & -1 & -1\end{bmatrix} = \begin{bmatrix}-0.5 & 2 & 0.5\end{bmatrix},$$
$$b = 0.5 - 1 = -0.5$$

И далее можем удостовериться, что откалиброванные значения весовых коэффициентов и смещения вполне подходят нашей обучающей выборке и в первом, и во втором случае.

Важно отметить, что полученное решение не является единственным или оптимальным. В самом деле, весовые коэффициенты $w = \begin{bmatrix}0 & 1 & 0\end{bmatrix}$ и смещение $b = 0$ при изменении выхода при p_1 на –1 тоже подходят для истинности обучающей выборки, но при этом состоят только из нулей и единиц. Правильный выбор обучающей выборки и ее способа кодирования, начальных случайных весов и смещения, как видно, существенно влияет на простоту и точность конечного решения.

Этот вывод важно всегда помнить при использовании искусственных нейронных сетей – в зависимости от начальных значений весовых коэффициентов и смещения, а также от способа минимизации ошибки, могут получаться разные параметры нейронной сети, хотя результат проверки на обучающей выборке при этом они могут выдавать одинаковый.

Кроме того, присутствует вопрос сходимости алгоритма поиска весовых коэффициентов, как в примере выше. Здесь этот вопрос не рассматривается, так как существуют соответствующие работы [1], в полной мере описывающие необходимое доказательство сходимости алгоритма обучения нейронных сетей с учителем.

Как известно, модели авторегрессии, да и любые регрессионные модели вообще, очень хорошо подходят для прогнозирования ВР, так как они описывают функциональные зависимости текущих значений ряда от предыдущих отсчетов. Но для построения даже самой простой модели АР требуется провести огромное количество работы, самая трудная из которых – это определить порядок модели АПРСС (p, d, q). Для верного прогноза нам требуется оценить целых три численных параметра порядка модели, а затем еще и проверить ее на адекватность.

Для задачи построения прогноза для заданного временного ряда нам требуется построить некоторую формальную регрессионную модель с неизвестными коэффициентами при условии минимизации функции ошибки (например, среднеквадратичного отклонения прогнозных точек). Существует множество конфигураций нейронных сетей, отвечающих данной задаче, однако нами была выбрана структура нейронной сети, называемая нелинейной авторегрессией NAR (Nonlinear AutoRegressive), схема которой приведена на рисунке 9.6.

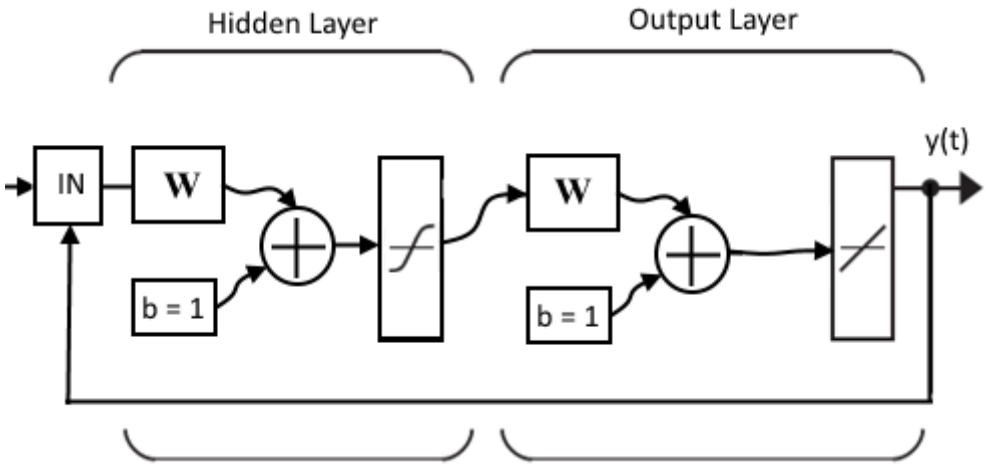

Рисунок 9.6 – Нейронная сеть NAR для прогнозирования временных рядов

Пусть прогноз ВР может быть описан в виде некоторой авторегрессионной зависимости. При этом форма АР порядка p может быть **нелинейной**, то есть мы не накладываем никаких ограничений на форму исходного ВР y и на вид функции f:

$$y(t) = f\left(y(t-1), y(t-2), ..., y(t-p)\right) \tag{9.6}$$

Для получения оптимального прогноза нам необходимо произвести «подгонку» функции (9.6) к известному ВР при условии минимизации среднеквадратичной ошибки отклонения (как и для обычных прогнозов АРПСС) . Модель прогноза (9.6) на основе данной нейронной сети (рис. 9.6) будет называться просто **нелинейной авторегрессией** (*Nonlinear Autoregressive* = **NAR**), а конечная прогнозирующая модель – **NAR-сетью**.

Из рисунка 6 видно, что NAR-сеть состоит из двух слоев: скрытого слоя и выходного слоя, имеет обратную связь с выхода на вход (рекуррентная кольцевая схема для возможности прогнозирования), смещение принято равным единице. Тренировка сети происходит на основе обучения с учителем, где обучающей выборкой служат отсчеты временного ряда. Функция активации скрытого слоя является лог-сигмоидой, функция активации выходного слоя является линейной.

Теперь у нас на руках есть исходный ВР, функция АР модели (9.6), нейронная сеть с заданными параметрами и принцип, по которому должна происходить минимизация ошибки обучения. Этого уже вполне достаточно для обучения нашей нейронной сети.

В таком случае алгоритм прогноза на основе NAR-сети будет выглядеть следующим образом:

1) Определяются параметры NAR-сети: число нейронов скрытого слоя и порядок *p* модели АР.
2) Часть отсчетов ВР (не менее 70%) используется для тренировки нейронной сети по принципу минимизации среднеквадратичной ошибки. Сами алгоритмы обучения могут быть разными.
3) Часть отсчетов ВР используется для проверки и валидации полученной модели по принципу ретроспективного прогнозирования.
4) В результате обучения нейронной сети будут адаптивно найдены коэффициенты АР модели (9.6).
5) Прогноз ВР происходит аналогично рекурсивным схемам, описанным для прогнозирования на основе АР моделей (лекция 8), за исключением того, что коэффициенты модели определяет сама нейронная сеть.

Прогноз на основе NAR-сети полностью идентичен прогнозированию на основе авторегрессионных моделей, за исключением таких ключевых факторов, как необходимость ручной оценки параметров модели и возможность создания нелинейностей в модели.

Преимуществами реализации NAR-сети являются ее простота, скорость обучения и высокая адаптивность – внутренние коэффициенты модели прогноза полностью зависят от самого ВР и возможностей нейронной сети.

Подобная простая схема NAR-сети в качестве входов использует сами значения исходного временного ряда, то есть нейронная сеть сопоставляет прошлые значения ряда с прогнозируемым наблюдением. Однако, как известно, исходный временной ряд может содержать различного рода шум, погрешности наблюдений и другие побочные вещи, что приводит к неустойчивости получаемого прогноза и самого процесса обучения сети.

Кроме того, такой прогноз NAR-сетью может быть только краткосрочным, так как не учитывает никаких характерных особенностей исходного временного ряда (амплитудные модуляции, циклы и т.д.)

Для решения этой проблемы NAR-сети дорабатывают до, так называемых, **NARX-сетей**. Рекуррентная динамическая нейронная сеть называется **нелинейной авторегрессионной сетью с внешними входами** (*Nonlinear Autoregressive with eXogenous inputs* = **NARX**), если ее моделью является расширенная АР порядка *d* модель вида:

$$y(t) = f\left(y(t-1), y(t-2), ..., y(t-d), x(t-1), x(t-2), ..., x(t-d)\right). \quad (9.7)$$

Сравнивая выражения (9.6) и (9.7) и рисунки 9.6 и 9.7 видно, что у нашей модели прогноза появился дополнительный вход, то есть теперь мы можем повлиять на процесс обучения нейронной сети и, соответственно, на получаемый прогноз.

Что же использовать в качестве входа $x(t)$? Вспомним, что проблемой нашей NAR-сети было искажение прогноза из-за шума в исходном ряде. Теперь вспомним из лекции 8 схему коррекции прогнозов АР моделей на основе новых наблюдений, и она будет почти один-в-один соответствовать выражению (9.7), если приравнять $x(t)$ – модели АР, а $y(t)$ – отсчетам ВР, включая и новые поступающие наблюдения.

По этой причине NARX-сети **не являются** чистой моделью прогноза ВР, они являются скорее **моделью коррекции первичного прогноза** $x(t)$ по отсчетам ВР $y(t)$, на основе которого строилась эта модель, при условии минимизации среднеквадратичной ошибки прогнозирования.

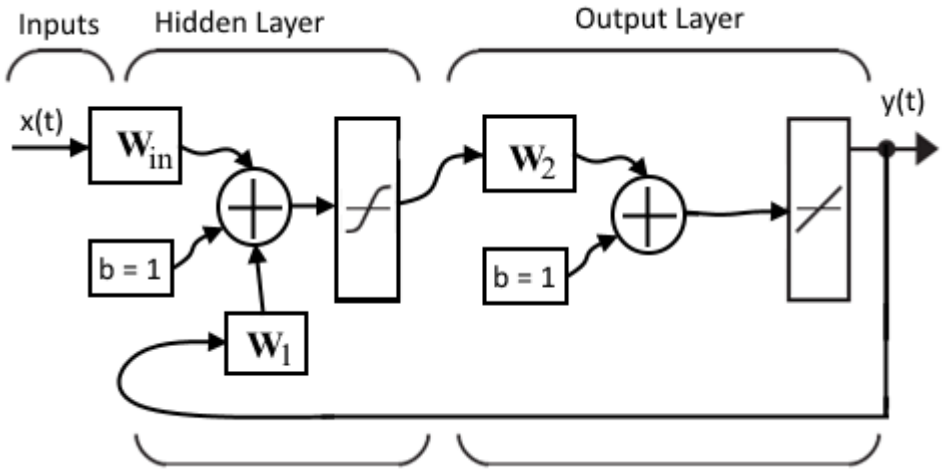

Рисунок 9.7 – Нейронная сеть NARX для коррекции базового прогноза $x(t)$

Таким образом, алгоритм прогноза будет выглядеть по-другому:

1) Для заданного ВР $y(t)$ строится его **упрощенная модель без шума** $x(t)$ с помощью любых алгоритмов декомпозиции/анализа ВР (например, просто в виде полиномиального тренда).
2) Для NARX-сети определяются ее параметры: число нейронов скрытого слоя и порядок модели АР (d) (лаг).
3) Происходит обучение нейронной NARX сети с учителем на основе отсчетов упрощенной модели ряда $x(t)$, выступающих в качестве входа NARX-модели, с коррекцией прогноза на основе отсчетов самого ВР $y(t)$, с задержкой d.
4) Прогноз на основе обученной NARX-сети строится следующим образом. С помощью упрощенной модели прогнозируют одиночный оценочный прогноз $x(t)$ на один шаг вперед, после чего NARX-сеть корректирует его с помощью выражения (9.7). Скорректированный прогноз $y(t)$ затем добавляется к исходному ВР. Затем происходит оценочный прогноз следующего шага и его коррекция уже по $y(t)$. Это процесс повторяется рекурсивно, пока не будет произведен прогноз в целом на заданное число шагов.

Таким образом, можно сделать вывод, что нейронные сети вполне применимы, как для прогнозирования ВР на основе NAR-сетей, так и для коррекции оценочного прогноза на основе NARX-сетей. **Преимуществами** использования нейронных сетей в подобной методике является их высокая универсальность и адаптивность, так как они не требуют от прогнозируемых ВР никаких особых свойств, а также огромные возможности вариации процесса обучения и формирования самих нейронных сетей. Так как сама структура нейронных сетей может существенно изменяться по ходу ее создания и обучения, это дает огромные возможности по самой тонкой настройке точности, направлению и скорости прогноза, которая не достижима в других формальных методах прогноза. **Недостатком** нейронных сетей являются их высокие вычислительные затраты, необходимые на обучение и валидацию. Кроме того, высокая вариабельность процесса построения моделей (9.6) и (9.7) может приводить к невоспроизводимым полученным результатам, то есть две нейронные сети с одинаковыми параметрами, но даже с очень малыми отклонениями в процедуре их обучения, могут давать совершенно разные прогнозы.

Еще отдельно стоит отметить особую роль NARX-моделей для прогнозирования. Мы рассматривали алгоритм коррекции прогноза, для которого оценочный прогноз $x(t)$ строился на основе самого ВР $y(t)$. Но для некоторых ВР известны **функциональные зависимости самих процессов**, что порождают данную выборку ряда. В этом случае NARX-сеть будет работать, как чистая прогнозирующая схема, у которой входом является модель процесса $x(t)$, породившего ряд $y(t)$, и нейронная сеть ищет зависимость между ними, которую затем можно будет использовать для прогноза.

Часть 3. Коррекция прогноза на основе ассимиляции данных

До сих пор мы обсуждали схемы коррекции прогноза, которые опирались только на сами модели и выражения, с помощью которых и строился данный прогнозирующий алгоритм. То есть схемы коррекции имели вид прогнозирующих функций, но с весовыми коэффициентами коррекции прогноза. Тем не менее, есть области науки, в которых прогнозирующая модель построена не для временного ряда, который является одномерным вектором наблюдений, а для самого многомерного процесса, породившего наблюдаемый ВР. Тогда прогноз происходит по этой многомерной модели, а его коррекция должна происходить по новым наблюдениям порожденного временного ряда.

Методы, которые как раз занимаются коррекцией многомерного прогноза по его одномерному (реже, двухмерному) ряду наблюдений называются **методами ассимиляции данных** (*Data Assimilation*). Так как в русском языке термин «ассимиляция данных» имеет различные значение, мы будем использовать просто аббревиатуру DA.

В целом, по строгому определению, **методы семейства DA** занимаются установлением однозначного соответствия между многомерным пространством решений некоторой прогностической модели и одномерным вектором имеющихся для этой системы наблюдений с учетом ошибок, возникающих, как в самой модели, так и в наблюдениях.

Исходная стохастическая модель системы формируется в виде следующей системы уравнений:

$$\begin{cases} x_{k+1} = M\left(x_k\right) + w_k, \\ y_k = H\left(x_k\right) + v_k \end{cases} \tag{9.8}$$

где x_k – известное состояние модели в момент времени t_k; x_{k+1} – состояние модели в следующий момент времени; M – оператор или функция перехода, определяющая эволюцию системы со временем; w_k – шум, присутствующий внутри модели; y_k – вектор наблюдений системы; H – оператор наблюдений, связывающий многомерное состояние системы с одномерным вектором наблюдений; v_k – ошибка наблюдений. Также предполагается, что обе эти ошибки являются независимыми случайными величинами. Модель M должна быть известна и определена, способ преобразования многомерной модели к порожденному ряду наблюдений H – тоже.

Первое уравнение в (9.8) называется **уравнением прогноза**, а второе – **уравнением наблюдений**. Соответственно, весь алгоритм DA разделяется на два этапа: **шаг анализа** и **шаг прогноза**.

На этапе анализа на основе имеющегося состояния системы x_k^f строится скорректированное состояние x_k^a, опирающееся на значение наблюдений y_k для момента времени t_k:

$$x_k^a = x_k^f + K\left(y_k - H\left(x_k^f\right)\right), \tag{9.9}$$

где K – коэффициент передачи, $e = y_k - H\left(x_k^f\right)$ – расчетная ошибка между известными наблюдениями и наблюдениями, найденными из нашей модели.

Задачей такого анализа в этом случае является минимизация этой ошибки e с учетом всех имеющихся факторов. В зависимости от того, как минимизируется эта ошибка, и различаются используемые методики DA.

На этапе прогноза строится состояние системы в следующий момент времени t_{k+1}, с учетом всех ошибок в исходной системе (9.8) и с учетом тех корректировок, что были внесены на шаге анализа (9.9):

$$x_{k+1}^{f} = M\left(x_{k}^{a}\right) \tag{9.10}$$

Таким образом, будет происходить **коррекция прогноза**: модель дает некоторый прогноз, порождая временной ряд, который сравнивается с имеющейся выборкой наблюдений, после чего по ним корректируется само многомерное состояние модели, которое нам пригодится на следующем шаге прогноза. Схема такой коррекции представлена на рисунке 9.8:

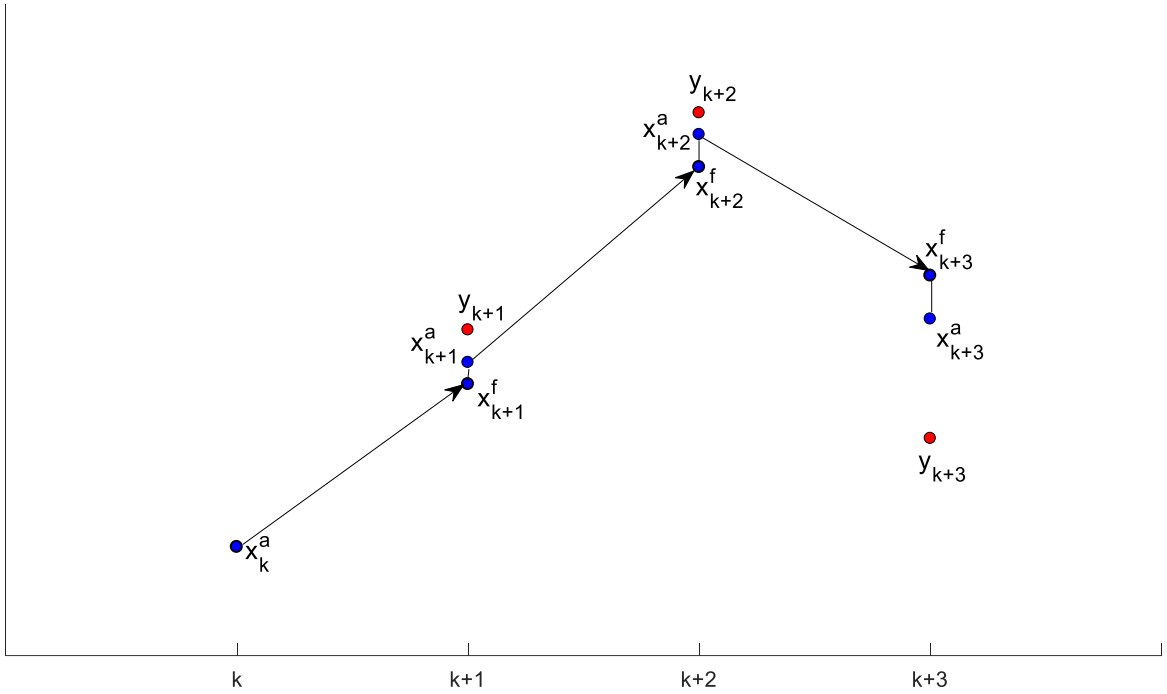

Рисунок 9.8 – Приближенная схема коррекции прогноза с помощью DA

В исходной постановке задачи, модель является линейной и операторы из (9.8) представлены в виде матриц, т.е. $M(x_k)=M\cdot x_k$ и $H(x_k)=H\cdot x_k$. Также считается, что шумы w_k и v_k являются выборками из нормального белого шума с заранее известными значениями ковариационной матрицы ошибок Q_k и R_k

соответственно. В этом случае **коэффициент передачи *K*** рассчитывается по формуле:

$$K_k^* = P_k^f H_k^T \left(H_k P_k^f H_k^T + R_k \right)^{-1} \tag{9.11}$$

где $P_k^f = M_k \left(I - K_{k-1}^* H_{k-1} \right) P_{k-1}^f M_k^T + Q_k$ – ковариационная матрица ошибок прогноза. Заметьте, что эти выражения очень похожи на коррекцию регрессионного прогноза, которую мы изучали раньше. В этом нет ничего удивительного – оба они строятся по принципу минимизации ошибки отклонений ряда от фактических наблюдений.

Параметр K^* в выражении (9.11) часто называют **коэффициентом передачи Калмана** (*Kalman gain*). Поэтому и семейство методик ***DA*** называют различными **фильтрами Калмана**.

Для линейного случая построения модели в виде матрицы M и отображения на временной ряд H, **алгоритм коррекции прогноза** на основе DA очень прост и будет выглядеть так. Пусть известно исходное состояние модели x_0^f и исходная ковариационная матрица ошибок P_0^f . Для шага t_k повторяется следующая последовательность действий:

1) Шаг анализа. Рассчитывается коэффициент передачи Калмана согласно (9.11).
2) Рассчитывается коррекция состояния модели на основе (9.9).
3) Находится ковариационная матрица ошибок анализа:

$$P_k^a = \left(I - K_k^* H_k \right) P_k^f \tag{9.12}$$

4) Шаг коррекции прогноза. Рассчитывается скорректированный прогноз на основе выражения (9.10).
5) Рассчитывается новая ковариационная матрица ошибок, необходимая для следующего шага:

$$P_{k+1}^f = M_{k+1} P_k^a M_{k+1}^T + Q_k \tag{9.13}$$

Важно отметить, что временные шаги t_k , указанные в алгоритме, **не обязательно должны совпадать с отсчетами временной сетки ряда**. То есть производить коррекцию прогноза можно, например, каждые 6 наблюдений, или корректировать суточный прогноз помесячно и т.д.

На практике в большинстве случаев операторы *M* и *H* не могут быть представлены в виде матриц. Чаще всего в их роли выступают целые программные функции и алгоритмы, которые для некоторого входного состояния многомерной модели сопоставляют следующее состояние (*M*) или порождают отсчет ВР (*H*). Еще более правдоподобно, когда и характеристики шумов нам будут неизвестны, то есть, ковариационных матриц ошибок тоже нет. В этом случае используется семейство методик **с фильтром Калмана по ансамблю** (*Ensemble Kalman Filter* = **EnKF**).

Из всего семейства этих методик, которые исчисляются десятками, мы рассмотрим только наиболее универсальный, называемый **методом стохастического фильтра Калмана по ансамблю** (*The Stochastic Ensemble Kalman Filter* = **SEnKF**). Алгоритм этого метода выглядит следующим образом: для каждого момента времени t_k повторяется следующая последовательность действий

1) Для фиксированной точки y_k наблюдения создается ансамбль наблюдений:

$$y_i = y_k + u_i, \ \sum_i u_i = 0 \tag{9.14}$$

при этом ковариационная матрица ошибок оценивается как

$$R_k = \frac{1}{m-1}\sum_i u_i u_i^T . \tag{9.15}$$

2) Рассчитывается коэффициент передачи (9.11), где произведения матриц заменяются статистической оценкой в виде:

$$P_k^f H_k^T = \frac{1}{m-1}\sum_i \left(x_i^f - \bar{x}^f\right)\left[H\left(x_i^f\right) - \bar{y}^f\right]^T$$
$$H_k P_k^f H_k^T = \frac{1}{m-1}\sum_i \left(H\left(x_i^f\right) - \bar{y}^f\right)\left[H\left(x_i^f\right) - \bar{y}^f\right]^T \quad (9.16)$$

где $\bar{y}^f = \frac{1}{m}\sum_i H\left(x_i^f\right)$.

3) Согласно (9.9) строится ансамбль скорректированных состояний x_i^a.

4) Строится прогноз по скорректированным значениям из ансамбля согласно (9.10) и из него находится средний прогноз

$$\bar{x}^f = \frac{1}{m}\sum_i x_i^f \ . \quad (9.17)$$

5) Рассчитывается ковариационная матрица ошибок прогноза для следующего шага

$$P^f = \frac{1}{m-1}\sum_i \left(x_i^f - \bar{x}^f\right)\left(x_i^f - \bar{x}^f\right)^T \quad (9.18)$$

Таким образом, с помощью различных схем коррекции прогноза на основе ассимиляции данных DA можно повышать точность прогноза по новым наблюдениям отсчетов ВР, имея на руках многомерную модель эволюции процесса, породившего данный временной ряд.